# SELECTIVE DISSEMINATION OF INFORMATION

# FUNCTIONAL SPECIFICATIONS

PREPARED BY: FUTURE SYSTEM
INFORMATION SCIENCE DIVISION

ED MA

# Profile Dictionary

Search profiles are stored in a MINISIS database called SDIPRF. This database is updated using MINISIS ENTRY and MODIFY processors or it can also be updated in the QUERY processor by the EDIT command in SDI mode.

The following is the description of the fields in SDIPRF database:

| FIELD NAME | MNEMONIC | TAG | LEN | CHARACTERISTICS |
|---|---|---|---|---|
| USER NUMBER | USERNO | A010 | 6 | |
| JOB INFORMATION | JOBINF | A100 | 100 | REPEAT |
| JOB NUMBER | JOBNUM | A101 | 6 | BTREE (SDIJ) |
| EXPLANATION | EXPLAN | B010 | 100 | |
| PROFILE THRESHOLD WT | THRESH | B020 | 4 | |
| SEARCH | SEARCH | B100 | 108 | REPEAT |
| LINE NUMBER | LNO | B101 | 5 | NUMERIC |
| SEARCH LINE | LINE | B102 | 100 | |
| PATTERN WEIGHT | WEIGHT | B103 | 4 | |

The USERNO field contains the number of the users of the profile. Information about users such as name, address and other billing information is maintained through an auxiliary MINISIS database called SDIUSR. The ISN of SDIUSR is the user number. (This data base is used in the PRINT processor to print the user information along with the searched records.)

The JOBINF field is a repeating group containing one subfield JOBNUM. Additional information will be added to this repeating group at a later date if required. The JOBNUM subfield identifies information about which jobs the profile belongs to. It has an associated inverted file linked to this field. The fast access path is used by the REFORMAT processor to retrieve all the profiles in a specified job. The JOBNUM fast access path can be useful to select a set of profiles to be modified by the EDIT command in the QUERY processor.

The EXPLAN field contains a short explanatory note about the profile.

The threshold field (THRESH) allows user to define a default weight for a hit record.

The SEARCH repeating group contains 3 subfields which are LNO, LINE and WEIGHT. The LNO subfield contains physical line number of a search line. The LINE subfield contains the text of the search line in regular query language format. The last subfield WEIGHT is used to assign weight for a search pattern.

Below is the description of the fields in SDI user database (SDIUSR):

| FIELD NAME | MNEMONIC | TAG | LEN | CHARACTERISTICS |
|---|---|---|---|---|
| USER NAME | USER | U100 | 40 | BTREE(SDIU) |
| USER ADDRESS | ADDR | U200 | 100 | |
| USER PROFILE JOBNO | UJOBNO | U300 | 6 | |

The first two fields are self-explanatory. The UJOBNO field is used to assign a job number to a user so that several of his profiles can be combined into a single job when doing the SDI searching.

Please note that user can define more fields into this database so that extra information related to the user can be added to this database.

A profile consists of a number of search lines. Each search line contains a line number and a clause. The construction of a search line clause is the same as setting up the search questions in QUERY processor. The SDI search clause is composed of 2 parts. The first part is the same as regular search question used in QUERY. The optional second part is a weight assignment for a search pattern. Four different weighting scales can be assigned to a pattern and can be described as VIMP, IMP, LIMP and NIMP. Below is the examples of constructing a search line:

THRESHOLD: VIMP (very important term)

1. A100 CAT

If there is no weight assigned to this pattern, a threshold weight of VIMP will be assigned to the record if it matches the above pattern otherwise a weight of zero is assinged to this record if there is no match.

2. A100 MOUSE IMP (important term)

A weight of IMP will be assigned to the term if it matches the above pattern otherwise a weight of zero is assigned to this term.

Examples of the search profile would look like as follows:

Examples #1.

Profile Number: 432  For user number: 220

Explanation: ERGONOMICS

JOBNUM: 8

| LINE # | SEARCH LINE |
|---|---|
| 1 | A100 ETITL |
| 2 | A100 TITLEA |
| 3 | B100 PRODUCT |
| 4 | B100 TITLE |
| 5 | A100 WORKING ADJ CONDITION |
| 6 | B100 @CRANK |
| 7 | (1 OR 2 OR 3) AND (4 OR 5 OR 6) |

Example #2.

Profile Number: 433  For user number: 221

Explanation: ERGONOMICS

JOBNUM: 8

THRESHOLD: VIMP

| LINE # | SEARCH LINE | WEIGHT |
|---|---|---|
| 1 | A100 ETITL | VIMP |
| 2 | A100 TITLEA | LIMP |
| 3 | A100 WORKING ADJ CONDITION | NIMP |
| 4 | B100 @CRANK | IMP |
| 5 | (1 OR 2) AND (3 OR 4) | |

Please note the following symbols:

VIMP - very important term
IMP - important term
LIMP - less important term
NIMP - not important term

## Weighting Algorithm

1. ORS weight

```
If weight(x) + weight(y) >= Threshold weight
  resultant weight = threshold weight
else
  resultant weight = weight(x) + weight (y)
```

2. ANDS weight

```
resultant weight = weight(x) * weight(y) / threshold
```

Suppose, there are 4 records (ISNS) with terms matched with the patterns in the above example 2. The weighting assignment would look like as:

| LINE # | WEIGHT | ISN LIST |
|---|---|---|
| 1 | VIMP | 1,3,4,9 |
| 2 | LIMP | 9 |
| 3 | NIMP | 3,4 |
| 4 | IMP | 1,4,9 |

The following chart determines the weights for the logic line in ORS and ANDS operations:

| OR | VIMP | IMP | LIMP | NIMP |
|---|---|---|---|---|
| VIMP | VIMP | VIMP | VIMP | VIMP |
| IMP | VIMP | VIMP | VIMP | VIMP |
| LIMP | VIMP | VIMP | VIMP | LIMP |
| NIMP | VIMP | VIMP | LIMP | LIMP |

| AND | VIMP | IMP | LIMP | NIMP |
|---|---|---|---|---|
| VIMP | VIMP | IMP | LIMP | NIMP |
| IMP | IMP | IMP | LIMP | NIMP |
| LIMP | LIMP | LIMP | NIMP | NIMP |
| NIMP | NIMP | NIMP | NIMP | NIMP |

By using the above charts, the weights for the logic line can be calculated as follows:

| LINE # | ISN/WT LIST |
|---|---|
| 1 OR 2 | 1/VIMP,3/VIMP,4/VIMP,9/VIMP |
| 3 OR 4 | 1/VIMP,3/NIMP,4/VIMP,9/VIMP |
| 5 | 1/VIMP,3/NIMP,4/VIMP,9/VIMP |

Finally, a list of ISNS with weights can be found and user can list all the hit records in the order of weights.

## Database Searching

Once the profiles are stored in the MINISIS database (SDIPRF) without any syntax errors, they are used to search the converted database in the QUERY processor by initiating a new command called SUBMIT. This command allows a user to search one profile or a set of profiles depending on which parameters he selects.

The user can test the searches on-line. If he is not satisfied with a first trial, the processor allows him to modify the search profiles on-line by using the EDIT DB command which in turn would initiate the MODIFY processor. The user can re-submit the profile searching by using the SUBMIT command again. He could repeat the above processing again and again until he is satisfied with the searches.

Alternatively, he can test his SDI searches as if he was in QUERY. After all the searches have been compeleted, the entire search questions or a portion of them can be stored into the SDI profile database by an enhanced KEEP command.

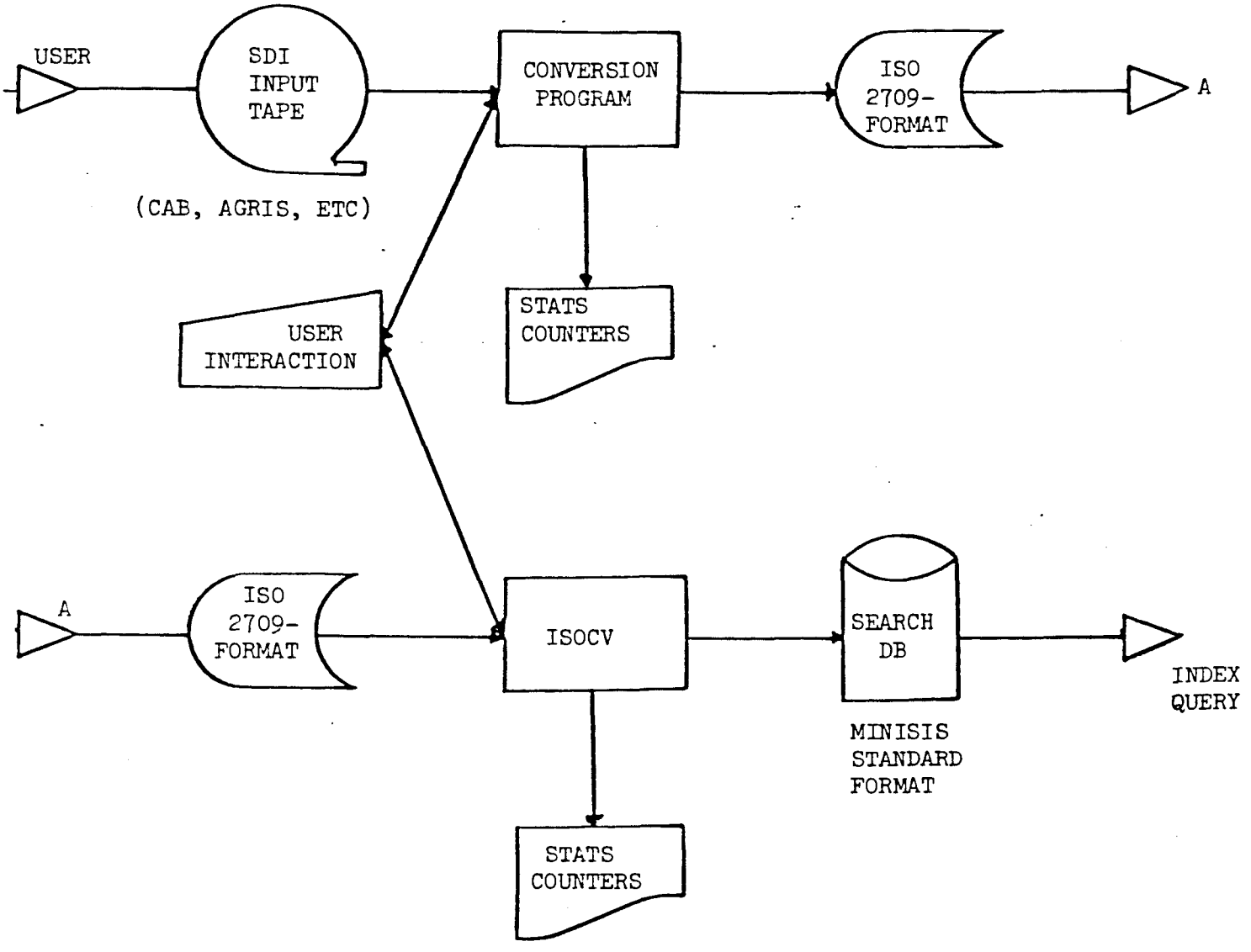
This processor also produces term counts report to the user. This report shows the details of the search questions plus the number of hits for each profile.

## Searched Record Reports

The details of the searched record reports will be produced in the PRINT processor using the hit file created in the Query processor.

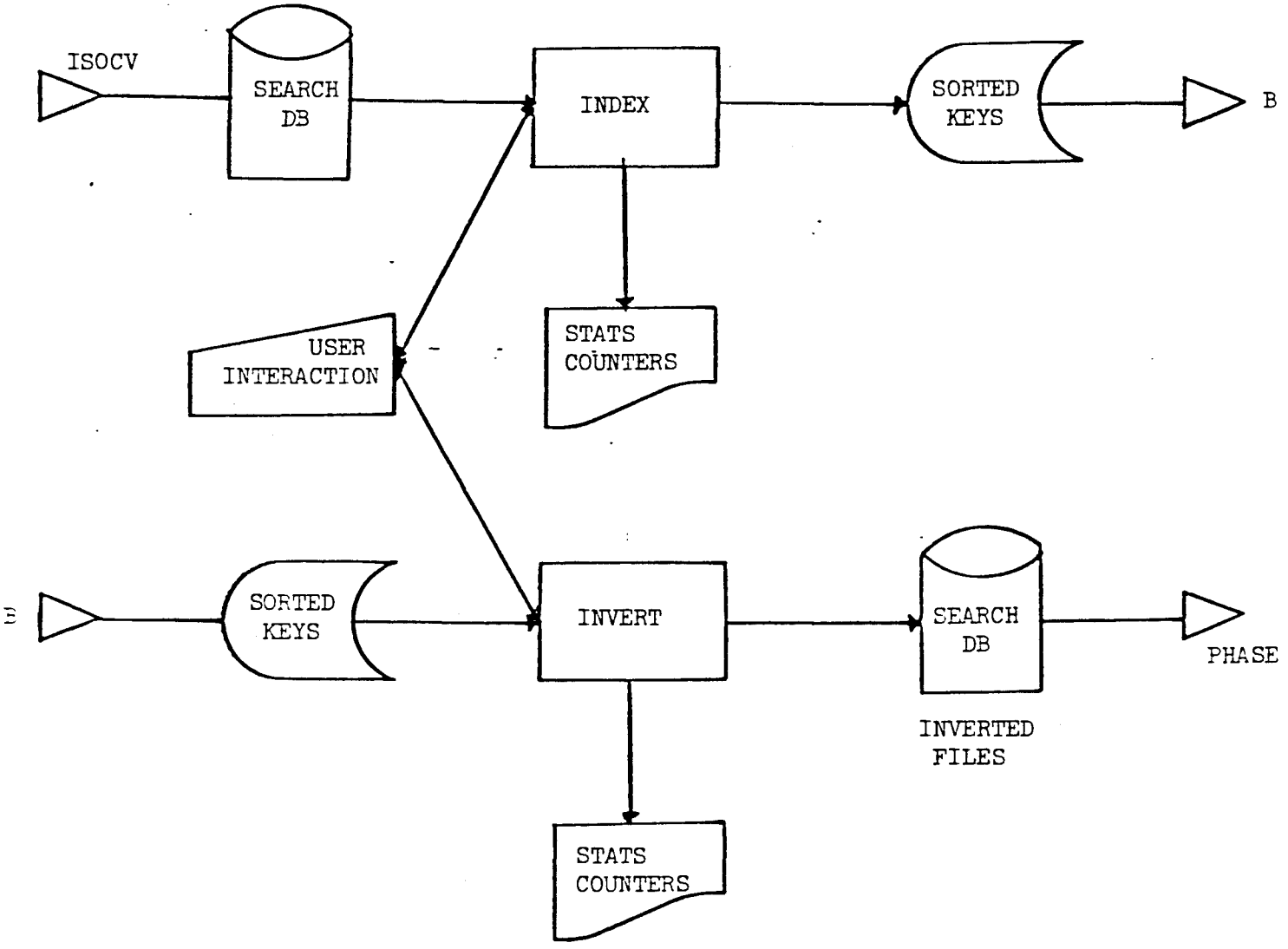
PHASE I

- INPUT CONVERSION



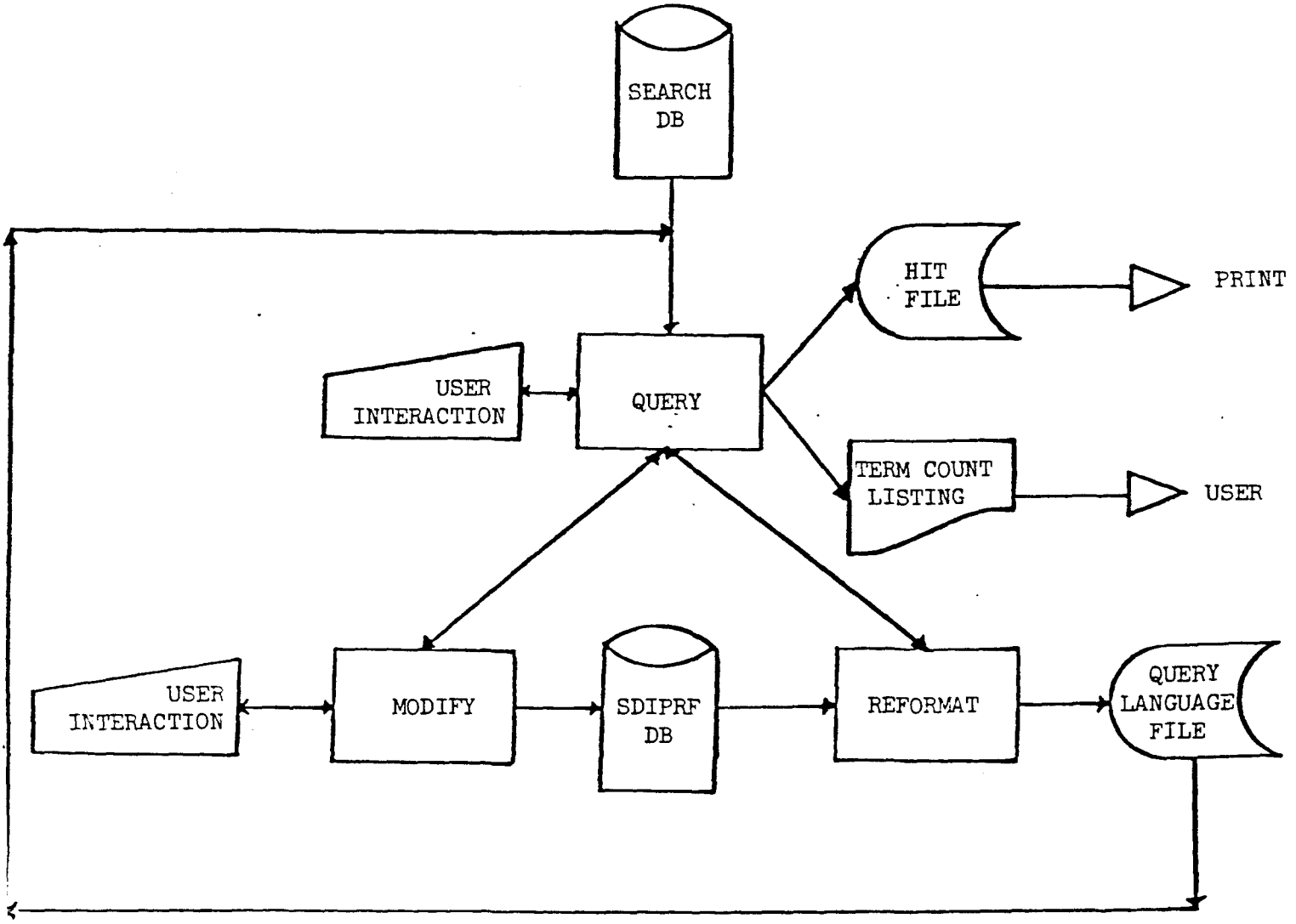
PHASE I

- INVERTED FILE CREATION



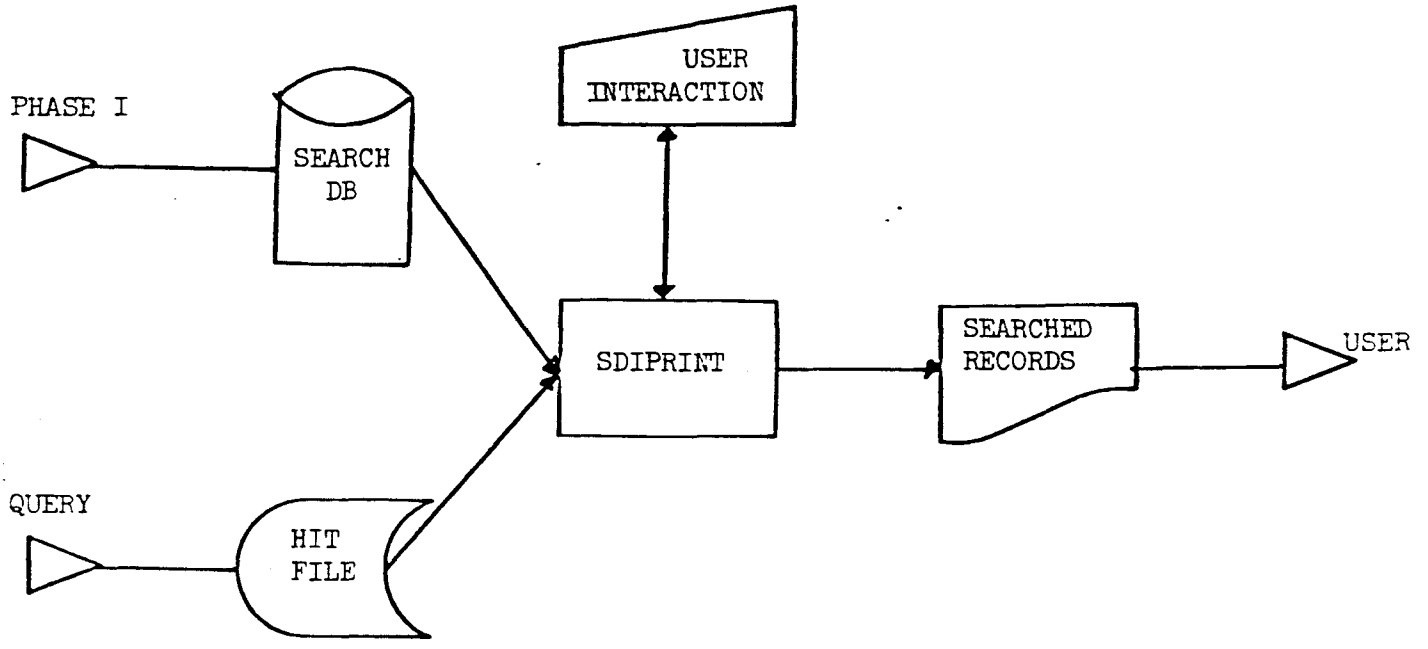
PHASE II

- USER PROFILE CREATION
- SDI SEARCHES



PHASE II

- SDI PRINT



(1) To allow the MODIFY processor to create a new record into the data base just like in the ENTRY processor. The following changes would be required:

- AUGMENT Intrinsic
  - add a new mode (REP) to this intrinsic to specify that the new record would be created even though it is a new one.

- MODIFY Processor
  - accept creating a non-existing record and call AUGMENT with REP mode. This must be done when called from the QUERY processor in the SDI mode.

(2) To provide syntax checking on search lines in the MODIFY processor.

This would require an EXIT in the MODIFY processor to check the syntax of the search line. This EXIT would perform the following functions:

- use the similar syntax table in EXTRACT or RESTRICT intrinsic to check the syntax of the search line.

- The syntax table should include the syntax checking for left truncation and adjacency searches. The new syntax for the above is as follows:

    a) A100 XYZ ADJ ABC or A100 XYZ : ABC
    Adjacency search

    b) A100 @XYZ
    Left truncation search

- All search line numbers have to be cross-reference checked to ensure that the number specified in the logic line is valid.

(3) To provide a query language file in editor-compatible format and a weight table which are used as input to QUERY processor.

A new processor (REFORMAT) would be required and would perform the following functions:

- convert a profile or set of profiles (retrieved from SDIPRF database) into a file in editor-compatible format.
- For each profile retrieved, the search line numbers would be re-sequenced in proper order and those appearing in the logic lines would be adjusted if necessary.
- create a weight table in EDS by line number and weight for use in RESTRICT intrinsic to assign weights to each searched record.

(4) To provide a routine in the QUERY processor to act as a driver to call MODIFY and REFORMAT processors dynamically.

This routine would perform the following functions:

- make use of the process handling capability available in MPE to activate the MODIFY or REFORMAT processors as required.

- be able to suspend any father's or son's processor if necessary so that at any one point in time, only one processor is activated.

- In order to facilitate QUERY to communicate with the MODIFY and REFORMAT processors via process handling capabilities, a small change in MODIFY would be required.

(5) To provide new commands in the QUERY processor for the SDI users.

(i) SETSDI {ON|OFF}

- allows user to run Query processor in SDI mode.

(ii) EDIT [dbname]

- allows user to modify the profile database on-line. This command would dynamically activate or create the MODIFY processor via a special routine.

(iii) SUBMIT {JOB=#/PROF=#} [HL={ON|OFF}] [,dbname]

- allows user to submit a profile or a set of profiles for searching by specifying a particular profile or job number. This command would dynamically create or activate the REFORMAT processor to retrieve the the profiles from SDIPRF database for SDI searches. An additional option is also to provide for user to highlight the searched records as required.

(iv) OUTPUT hitfile name

- to save the all the searched records to be used in SDIPRINT porcessor. The records are stored as Job Number, User Number, Profile Number, Weight scale, plus the highlighted terms.

(v) KEEP DB=dbname,PROF=#,JOB=#

- allows user to keep the entire search questions or a portion of them into the profile database.

(6) To provide 2 new searching capabilities in QUERY processor, they are as follows:

- Left Truncation Search

  A special inverted file would be required. It will contain 2 sets of directory entries; one is used for regular searching and the other for left truncation search. For this type of search, keys are stored and sorted in reversed order with pointers pointing to the same bit string as in the first set of directory entries. This kind of search can be accomplised by sequentially examining the keys in the secondary directory entries against the key supplied by the user.

- Adjacency Search

  Create a special type of inverted file with each key in the directory entry carrying two types of information. The first carries ISN information and stores as a bit string with each bit representing one ISN. The other is an integer string composed of several entries with each one carrying information required for adjacency search, such as ISN, TAG, field occurrance number and the word postion number relative to the beginning of the field. By using the information specified in the integer string entry, the adjacency search would be accomplished.

(7) To provide a feature of assigning weights to each searched record so that the records would be sorted in the order of weights.

To accomplish this, the following changes would be required:

- RESTRICT intrinsic would be changed to assign weights to each hit record and create an integer string (ISN/WEIGHT) in the intermediate work file. This string is used to evaluate the weighting for the logic line in the search questions.

- create a hit file with weight assigned to each ISN so that the file can be sorted by weight in SDIPRINT processor prior printing the searched records.

| Format: | Search ISN, | Job #, | User #, | Prof #, | Weight, | Terms |
|---|---|---|---|---|---|---|
| word: | 0 | 4 | 8 | 12 | 16 | 18 |

Please note that this file is sorted by job #, user #,prof #, weight and search ISN so that the searched records can be listed in the order of weights within a profile.

(8) To provide a facility to highlight the searched records in the SDIPRINT processor.

To implement the above feature the following changes would be required:

- change QUERY processor to create a hit file containing the information such as ISN, WEIGHT and PATTERN TERMS. This will enable SDIPRINT to list the searched records along with the pattern terms plus all the user information from user database.

- by means of a user THRESHOLD weight, a subset of the searched records can be printed. For example, user can request to print all those records with weights higher or lower than the THRESHOLD weight.

## Time Estimate for SDI Implementation

| | MANDAYS |
|---|---|
| 1. Develop detail specifications | 15 |
| 2. User profile creation<br>- allow MODIFY processor to create new record.<br>- add new mode to AUGMENT intrinsic to create new records.<br>- write new EXIT (used in the MODIFY processor) to check the syntax of search lines. | 5 |
| 3. Query language file creation<br>- write a new processor (REFORMAT) to convert profiles into editor-compatible format. | 4 |
| 4. Interprocess communication<br>- write a routine in the QUERY processor to call MODIFY and REFORMAT processors dynamically. | 3 |
| 5. Introduce new commands in the QUERY processor. They are:<br>SETSDI<br>EDIT dbname<br>SUBMIT<br>OUTPUT filename<br>KEEP (enhanced command) | 3 |
| 6. Implement left truncation search<br>- add new parameters required in the command to create a special type of inverted file in Datadef.<br>- add a new routine in Invert to create an extra directory entry for keys sorted in reversed order.<br>- accept new syntax for left truncation search in the RESTRICT and KEYS intrinsics.<br>- modify TREE intrinsic to retrieve the keys from the 2nd set of directory entries in the special inverted file. | 15 |

| | | |
|---|---|---|
| 7. | Implement adjacency search<br>- add new parameters required in the command to create a special inverted file in Datadef.<br>- add new parameters required in Index for special inversion.<br>- add a new routine in INVERT to create a special inverted file with keys having 2 posting strings.<br>- modify syntax table in EXTRACT and RESTRICT to accept a new syntax for adjacency search.<br>- add a new routine (INTEGER'MANIP) to manipulate integer strings plus performing boolean operations on them. | 20 |
| 8. | Implement weighting feature<br>- add a new routine in RESTRICT in performing the evaluation on 2 integer strings and create ISN-WEIGHT string in the RESTRICT work file. | 10 |
| 9. | Highlight the searched records<br>- add a routine in the Query processor to include the pattern terms information in the Query hit file. | 10 |
| 10. | System testing | 20 |
| 11. | Operational documentation | 4 |
| | | ------- |
| | Total | 109+10<br>or<br>109-10 |
| | approximately | 6<br>man months |